Cost-based Domain Filtering for Stochastic Constraint Programming

Roberto Rossi[1]
S. Armagan Tarim[2]
Brahim Hnich[3]
Steven D. Prestwich[1]

[1]Cork Constraint Computation Centre, University College Cork, Ireland
[2]Department of Management, Hacettepe University, Ankara, Turkey
[3]Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey

14^{th} International Conference of Principles and Practice of Constraint Programming

Outline

1. **Introduction**
   - Constraint Programming
   - Stochastic Constraint Programming
2. **Global Optimization Chance-Constraints**
   - Motivations
   - Expectation-based relaxation for stochastic variables
   - Relaxing the expected value problem
   - Cost-based filtering
   - Finding good feasible solutions
3. **Experiments**
   - Static Stochastic Knapsack Problem
   - Stochastic sequencing with release times and deadlines
4. **Conclusions**
   - Related Works
   - Conclusions

Abstract

Abstract

- **Cost-based filtering** is a novel approach that combines techniques from Operations Research and Constraint Programming to filter from decision variable domains values that do not lead to better solutions (Focacci et al. [2]).

Abstract

- **Cost-based filtering** is a novel approach that combines techniques from Operations Research and Constraint Programming to filter from decision variable domains values that do not lead to better solutions (Focacci et al. [2]).
- **Stochastic Constraint Programming** is a framework for modeling combinatorial optimization problems that involve uncertainty (Walsh [7]).

Abstract

- **Cost-based filtering** is a novel approach that combines techniques from Operations Research and Constraint Programming to filter from decision variable domains values that do not lead to better solutions (Focacci et al. [2]).
- **Stochastic Constraint Programming** is a framework for modeling combinatorial optimization problems that involve uncertainty (Walsh [7]).
- In this work, we show how to perform **cost-based filtering** for certain classes of **stochastic constraint programs**.

Abstract

- Our approach is based on a set of **known inequalities** borrowed from Stochastic Programming — a branch of OR concerned with modeling and solving problems involving uncertainty.

Abstract

- Our approach is based on a set of **known inequalities** borrowed from Stochastic Programming — a branch of OR concerned with modeling and solving problems involving uncertainty.
- We discuss **bound generation** and **cost-based domain filtering procedures** for
  - the **Static Stochastic Knapsack Problem**
  - A **Stochastic Job Sequencing Problem**

Abstract

- Our approach is based on a set of **known inequalities** borrowed from Stochastic Programming — a branch of OR concerned with modeling and solving problems involving uncertainty.
- We discuss **bound generation** and **cost-based domain filtering procedures** for
  - the **Static Stochastic Knapsack Problem**
  - A **Stochastic Job Sequencing Problem**
- Our results clearly show the **value** of the proposed approach over a pure scenario-based Stochastic Constraint Programming formulation both in terms of

Abstract

- Our approach is based on a set of **known inequalities** borrowed from Stochastic Programming — a branch of OR concerned with modeling and solving problems involving uncertainty.
- We discuss **bound generation** and **cost-based domain filtering procedures** for
  - the **Static Stochastic Knapsack Problem**
  - A **Stochastic Job Sequencing Problem**
- Our results clearly show the **value** of the proposed approach over a pure scenario-based Stochastic Constraint Programming formulation both in terms of
  - **explored nodes**
  - **run time**

Constraint Programming

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

Constraint Programming

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables

Constraint Programming

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables
- D is a function mapping each variable v_i to a domain $D(v_i)$ of values

Constraint Programming

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables
- D is a function mapping each variable v_i to a domain $D(v_i)$ of values
- C is a set of constraints

Constraint Programming

A slightly formal definition

A Constraint Satisfaction Problem is a triple $\langle V, D, C \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of variables
- D is a function mapping each variable v_i to a domain $D(v_i)$ of values
- C is a set of constraints

Sample CSP

- $V = \{x, y\}$
- $D(x) = \{1, 3, 4, 5\}$ $D(y) = \{4, 5, 8\}$
- $C = \{x + 3 = y\}$

A possible solution for the CSP is $x = 1$ and $y = 4$.

Constraint Programming

Strategy

Constraint Programming

Strategy

Constraint Programming proposes to solve CSPs by associating with each constraint a **filtering algorithm**.

Constraint Programming

Strategy

Constraint Programming proposes to solve CSPs by associating with each constraint a **filtering algorithm**.

Filtering Algorithm

A filtering algorithm removes from decision variable domains **values that cannot belong** to any solution of the CSP.

Constraint Programming

Strategy

Constraint Programming proposes to solve CSPs by associating with each constraint a **filtering algorithm**.

Filtering Algorithm

A filtering algorithm removes from decision variable domains **values that cannot belong** to any solution of the CSP.

Constraint Propagation

...is the **process** that repeatedly calls filtering algorithms until no new deduction can be made.

Constraint Programming

Strategy

CP interleaves...

Constraint Programming

Strategy

CP interleaves...

- **filtering algorithms**

Constraint Programming

Strategy

CP interleaves...

- **filtering algorithms**
- **a search procedure**
  - i.e. a backtracking algorithm

Constraint Programming

Strategy

CP interleaves...

- **filtering algorithms**
- **a search procedure**
  - i.e. a backtracking algorithm

during the search filtering algorithms are systematically applied when the domain of a variable is modified.

Global Constraints

Not only binary relations

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"
- In constraint programming is common to find constraints over a **non-predefined number of variables**

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"
- In constraint programming is common to find constraints over a **non-predefined number of variables**
  - `alldifferent`
  - `element`
  - `cumulative`
  - ...

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"
- In constraint programming is common to find constraints over a **non-predefined number of variables**
  - `alldifferent`
  - `element`
  - `cumulative`
  - ...
- These constraints are called **global constraints**

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"
- In constraint programming is common to find constraints over a **non-predefined number of variables**
  - `alldifferent`
  - `element`
  - `cumulative`
  - ...
- These constraints are called **global constraints**
  - they can be used in a variety of situations

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"
- In constraint programming is common to find constraints over a **non-predefined number of variables**
  - `alldifferent`
  - `element`
  - `cumulative`
  - ...
- These constraints are called **global constraints**
  - they can be used in a variety of situations
  - they are associated with powerful **filtering strategies**

Global Constraints

Not only binary relations

- In the former example the constraint in C was "**binary**"
- In constraint programming is common to find constraints over a **non-predefined number of variables**
  - `alldifferent`
  - `element`
  - `cumulative`
  - ...
- These constraints are called **global constraints**
  - they can be used in a variety of situations
  - they are associated with powerful **filtering strategies**
  - new **custom** global constraints can be defined

Optimization-oriented global constraints

Optimization-oriented global constraints (Focacci et al. [2]) embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint. This component provides three pieces of information:

Optimization-oriented global constraints

Optimization-oriented global constraints (Focacci et al. [2]) embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint. This component provides three pieces of information:

- (a) the optimal solution of the **relaxed problem**;

Optimization-oriented global constraints

Optimization-oriented global constraints (Focacci et al. [2]) embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint. This component provides three pieces of information:

- (a) the optimal solution of the **relaxed problem**;
- (b) the optimal value of this solution representing an **upper bound** on the original problem objective function;

Optimization-oriented global constraints

Optimization-oriented global constraints (Focacci et al. [2]) embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint. This component provides three pieces of information:

- (a) the optimal solution of the **relaxed problem**;
- (b) the optimal value of this solution representing an **upper bound** on the original problem objective function;
- (c) a *gradient function* **grad(**V**,**v**)**, which returns for each couple variable-value (V,v) an optimistic evaluation of the profit obtained if v is assigned to V.

Optimization-oriented global constraints

Optimization-oriented global constraints (Focacci et al. [2]) embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint. This component provides three pieces of information:

- (a) the optimal solution of the **relaxed problem**;
- (b) the optimal value of this solution representing an **upper bound** on the original problem objective function;
- (c) a *gradient function* **grad(**V**,**v**)**, which returns for each couple variable-value (V,v) an optimistic evaluation of the profit obtained if v is assigned to V.

These pieces of information are exploited both for **propagation** purposes and for **guiding the search**.

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of **decision** variables

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of **decision** variables
- $S = \{s_1, \ldots, s_n\}$ is a set of **stochastic** variables

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of **decision** variables
- $S = \{s_1, \ldots, s_n\}$ is a set of **stochastic** variables
- D is a function mapping each variable to a **domain** of potential values

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of **decision** variables
- $S = \{s_1, \ldots, s_n\}$ is a set of **stochastic** variables
- D is a function mapping each variable to a **domain** of potential values
- P is a function mapping each variable in S to a **probability distribution** for its associated domain

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of **decision** variables
- $S = \{s_1, \ldots, s_n\}$ is a set of **stochastic** variables
- D is a function mapping each variable to a **domain** of potential values
- P is a function mapping each variable in S to a **probability distribution** for its associated domain
- C is a set of **constraints**

Constraint Programming in Optimization under Uncertainty

A slightly formal definition

A Stochastic Constraint Satisfaction Problem is a 6-tuple $\langle V, S, D, P, C, \theta \rangle$.

- $V = \{v_1, \ldots, v_n\}$ is a set of **decision** variables
- $S = \{s_1, \ldots, s_n\}$ is a set of **stochastic** variables
- D is a function mapping each variable to a **domain** of potential values
- P is a function mapping each variable in S to a **probability distribution** for its associated domain
- C is a set of **constraints**
- θ_h is a **threshold probability** associated with chance-constraint h

Constraint Programming in Optimization under Uncertainty

- A stochastic CSP consists of a number of **decision stages**

Constraint Programming in Optimization under Uncertainty

- A stochastic CSP consists of a number of **decision stages**
- A **decision stage** is a pair $\langle V_i, S_i \rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables

Constraint Programming in Optimization under Uncertainty

- A stochastic CSP consists of a number of **decision stages**
- A **decision stage** is a pair $\langle V_i, S_i \rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables
- In a 1**-stage stochastic CSP**, V and S are partitioned into disjoint sets, V_1 and S_1, and we consider a single stages, $\langle V_1, S_1 \rangle$

Constraint Programming in Optimization under Uncertainty

- A stochastic CSP consists of a number of **decision stages**
- A **decision stage** is a pair $\langle V_i, S_i \rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables
- In a 1**-stage stochastic CSP**, V and S are partitioned into disjoint sets, V_1 and S_1, and we consider a single stages, $\langle V_1, S_1 \rangle$
- To solve a 1-stage stochastic CSP an assignment to the variables in V_1 must be found such that, given random values for S_1 the hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios

Constraint Programming in Optimization under Uncertainty

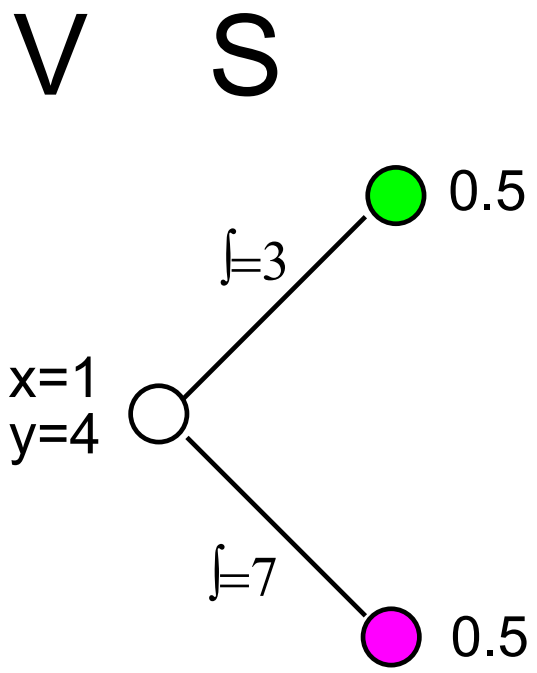
- A stochastic CSP consists of a number of **decision stages**
- A **decision stage** is a pair $\langle V_i, S_i \rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables
- In a 1**-stage stochastic CSP**, V and S are partitioned into disjoint sets, V_1 and S_1, and we consider a single stages, $\langle V_1, S_1 \rangle$
- To solve a 1-stage stochastic CSP an assignment to the variables in V_1 must be found such that, given random values for S_1 the hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios
- In general, the solution of an m**-stage stochastic CSP** is represented by means of a **policy tree** (Walsh [6])

Constraint Programming in Optimization under Uncertainty

Sample Stochastic CSP

- $V = \{x, y\}$
- $S = \{\int\}$
- $D(x) = \{1, 3, 4, 5\}$ $D(y) = \{4, 5, 8\}$
- $P(\int) = \{3(50\%), 7(50\%)\}$
- $C = \{x + \int = y\}$
- $\theta = 50\%$

A possible solution for the CSP is $x = 1$ and $y = 4$



Constraint Programming in Optimization under Uncertainty

- Let $\mathcal{S}$ denote the **space of policy trees** representing all the solutions of a stochastic CSP

Constraint Programming in Optimization under Uncertainty

- Let $\mathcal{S}$ denote the **space of policy trees** representing all the solutions of a stochastic CSP
- We may be interested in finding a feasible solution, i.e. a policy tree $s \in \mathcal{S}$, that **maximizes** (or minimizes) the value of a given objective function $f(\cdot)$ over the stochastic variables S (edges of the policy tree) and over a subset $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree)

Constraint Programming in Optimization under Uncertainty

- Let $\mathcal{S}$ denote the **space of policy trees** representing all the solutions of a stochastic CSP
- We may be interested in finding a feasible solution, i.e. a policy tree $s \in \mathcal{S}$, that **maximizes** (or minimizes) the value of a given objective function $f(\cdot)$ over the stochastic variables S (edges of the policy tree) and over a subset $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree)
- A **Stochastic COP** is then defined in general as $\max_{s \in \mathcal{S}} f(s)$

Constraint Programming in Optimization under Uncertainty

Stochastic Constraint Programming

Also in Stochastic Constraint Programming (SCP) we have

Constraint Programming in Optimization under Uncertainty

Stochastic Constraint Programming

Also in Stochastic Constraint Programming (SCP) we have

- constraints

Constraint Programming in Optimization under Uncertainty

Stochastic Constraint Programming

Also in Stochastic Constraint Programming (SCP) we have

- constraints
- filtering Algorithms

In contrast to CP, in SCP constraints divide into

Constraint Programming in Optimization under Uncertainty

Stochastic Constraint Programming

Also in Stochastic Constraint Programming (SCP) we have

- constraints
- filtering Algorithms

In contrast to CP, in SCP constraints divide into

- hard constraints

Constraint Programming in Optimization under Uncertainty

Stochastic Constraint Programming

Also in Stochastic Constraint Programming (SCP) we have

- constraints
- filtering Algorithms

In contrast to CP, in SCP constraints divide into

- hard constraints
- chance-constraints

Constraint Programming in Optimization under Uncertainty

Stochastic Constraint Programming

Also in Stochastic Constraint Programming (SCP) we have

- constraints
- filtering Algorithms

In contrast to CP, in SCP constraints divide into

- hard constraints
- chance-constraints

Global Chance-Constraints

Perhaps the most interesting aspect of SCP is that the concept of **global constraint** can be also adopted in a stochastic environment, thus leading to

- Global Chance-Constraints (Rossi et al., 2008)

Static Stochastic Knapsack

SSKP

- a set K of items

Static Stochastic Knapsack

SSKP

- a set K of items
- each item i has an expected reward r_i

Static Stochastic Knapsack

SSKP

- a set K of items
- each item i has an expected reward r_i
- the size $\mathcal{W}_i$ of each item is **not known** at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$

Static Stochastic Knapsack

SSKP

- a set K of items
- each item i has an expected reward r_i
- the size $\mathcal{W}_i$ of each item is **not known** at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$
- a subset of k items has to be chosen, given a knapsack of size q into which to fit the items

Static Stochastic Knapsack

SSKP

- a set K of items
- each item i has an expected reward r_i
- the size $\mathcal{W}_i$ of each item is **not known** at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$
- a subset of k items has to be chosen, given a knapsack of size q into which to fit the items
- A per unit penalty of c has to be paid for **exceeding the capacity** of the knapsack

Static Stochastic Knapsack

SSKP

- a set K of items
- each item i has an expected reward r_i
- the size $\mathcal{W}_i$ of each item is **not known** at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$
- a subset of k items has to be chosen, given a knapsack of size q into which to fit the items
- A per unit penalty of c has to be paid for **exceeding the capacity** of the knapsack
- maximize the **expected profit** = total reward - penalty

Static Stochastic Knapsack

SSKP

We model the SSKP as a **one-stage** Stochastic COP, the recourse problem RP(SSKP) can be formulated as

Objective:

$$\max \left\{ \sum_{i=1}^{k} r_i X_i - c\mathbb{E}\left[\sum_{i=1}^{k} \mathcal{W}_i X_i - q \right]^+ \right\}$$

Decision variables:

(1) $X_i \in \{0, 1\}$ $\forall i \in 1, \ldots, k$

Stochastic variables:

$\mathcal{W}_i \rightarrow$ item i weight

Figure: RP(SSKP). Note that $[y]^+ = \max\{y, 0\}$.

Static Stochastic Knapsack

According to the approach described in Tarim et al. [6] we compile RP(SSKP) into a **deterministic equivalent** CP model using a **scenario-based** representation.

Objective:

$$\max \left\{ \sum_{i=0}^{k} r_i X_i - c \left[\sum_{j=1}^{n} Z_j \Pr\{j\} \right] \right\}$$

Constraints:

(1) $Z_j \geq \sum_{i=1}^{k} \mathcal{W}_i^j X_i - q \qquad \forall j \in 1, \ldots, n$

Decision variables:

(2) $X_i \in \{0, 1\} \qquad \forall i \in 1, \ldots, k$

(3) $Z_j \in [0, \sum_{i=1}^{k} \mathcal{W}_i^j] \qquad \forall j \in 1, \ldots, n$

Figure: DetEquiv(RP(SSKP)). $\Pr\{j\}$ is the probability of scenario $j \in \{1, \ldots, n\}$. Note that $\sum_{j=1}^{n} \Pr\{j\} = 1$.

Static Stochastic Knapsack

Consider 5 items,

Static Stochastic Knapsack

Consider 5 items,

- item **rewards** r_i are $\{10, 15, 20, 5, 25\}$.

Static Stochastic Knapsack

Consider 5 items,

- item **rewards** r_i are $\{10, 15, 20, 5, 25\}$.
- The discrete **probability distribution** functions $f(i)$ for the weight of item $i = 1, \ldots, 5$ are respectively,
  $f(1) = \{10(0.5), 8(0.5)\}$, $f(2) = \{10(0.5), 12(0.5)\}$,
  $f(3) = \{9(0.5), 13(0.5)\}$, $f(4) = \{4(0.5), 6(0.5)\}$,
  $f(5) = \{12(0.5), 15(0.5)\}$.

Static Stochastic Knapsack

Consider 5 items,

- item **rewards** r_i are $\{10, 15, 20, 5, 25\}$.
- The discrete **probability distribution** functions $f(i)$ for the weight of item $i = 1, \ldots, 5$ are respectively,
  $f(1) = \{10(0.5), 8(0.5)\}$, $f(2) = \{10(0.5), 12(0.5)\}$,
  $f(3) = \{9(0.5), 13(0.5)\}$, $f(4) = \{4(0.5), 6(0.5)\}$,
  $f(5) = \{12(0.5), 15(0.5)\}$.
- The figures in parenthesis represent the probability that an item takes a certain weight.

Static Stochastic Knapsack

Consider 5 items,

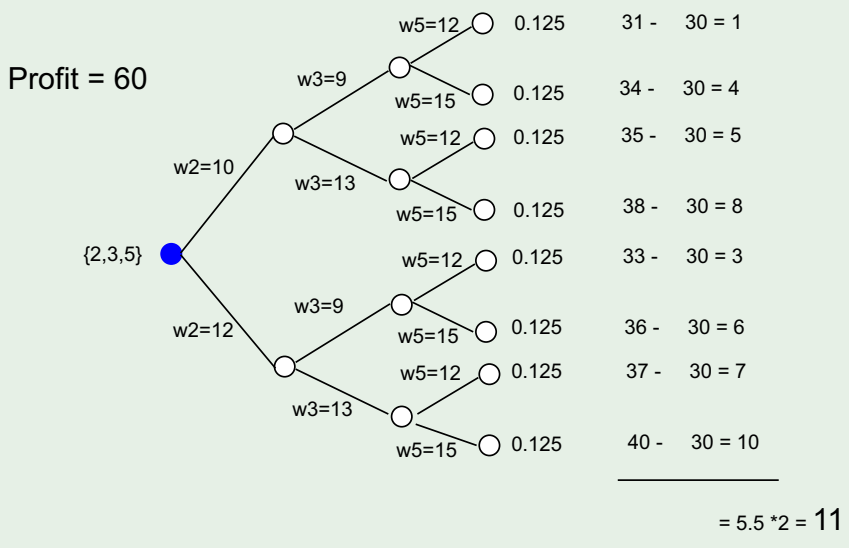
- item **rewards** r_i are $\{10, 15, 20, 5, 25\}$.
- The discrete **probability distribution** functions $f(i)$ for the weight of item $i = 1, \ldots, 5$ are respectively,
  $f(1) = \{10(0.5), 8(0.5)\}$, $f(2) = \{10(0.5), 12(0.5)\}$,
  $f(3) = \{9(0.5), 13(0.5)\}$, $f(4) = \{4(0.5), 6(0.5)\}$,
  $f(5) = \{12(0.5), 15(0.5)\}$.
- The figures in parenthesis represent the probability that an item takes a certain weight.
- The other problem parameters are $c = 2$, $q = 30$.

Static Stochastic Knapsack

- The **optimal solution** of the recourse problem selects items $\{2, 3, 5\}$ and has a value of RP(SSKP)=49.

Static Stochastic Knapsack

- The **optimal solution** of the recourse problem selects items $\{2, 3, 5\}$ and has a value of RP(SSKP)=49.
- This solution can be obtained by solving a **deterministic equivalent** conventional constraint program obtained by employing a **scenario-based** representation [6].



Motivations

- Solving stochastic constraint programs is **computationally a challenging task**. In (Walsh [7]), the computational complexity — membership in **PSPACE** — of these models is discussed.

Motivations

- Solving stochastic constraint programs is **computationally a challenging task**. In (Walsh [7]), the computational complexity — membership in **PSPACE** — of these models is discussed.
- In Tarim et al. [6], the authors proposed a standard way of **compiling** down these models into **conventional** (non-stochastic) **CP models** that can be solved by any available commercial software.

Motivations

- Solving stochastic constraint programs is **computationally a challenging task**. In (Walsh [7]), the computational complexity — membership in **PSPACE** — of these models is discussed.
- In Tarim et al. [6], the authors proposed a standard way of **compiling** down these models into **conventional** (non-stochastic) **CP models** that can be solved by any available commercial software.
- Of course this approach has **a price** since the number of scenarios that need to be considered in order to fully represent the problem **grows exponentially** with the number of decision stages in the problem.

Motivations

- A possible way to overcome this difficulty is to **reduce the number of scenarios** considered by **sampling** them, but this obviously affects the completeness of the model.

Motivations

- A possible way to overcome this difficulty is to **reduce the number of scenarios** considered by **sampling** them, but this obviously affects the completeness of the model.
- Another possibility consists instead in developing **specialized and efficient filtering strategies**. For this purpose **global chance-constraints** have been proposed in Rossi et al. [4].

Motivations

- A possible way to overcome this difficulty is to **reduce the number of scenarios** considered by **sampling** them, but this obviously affects the completeness of the model.
- Another possibility consists instead in developing **specialized and efficient filtering strategies**. For this purpose **global chance-constraints** have been proposed in Rossi et al. [4].
- In this work, by creating a parallel with Focacci et al. [2], we present **optimization-oriented global chance-constraints** as a way of enhancing the solving process of stochastic constraint programs.

Optimization-oriented global chance-constraints

A **global optimization chance-constraint** provides the same three pieces of information provided by optimization-oriented global constraints.

Optimization-oriented global chance-constraints

A **global optimization chance-constraint** provides the same three pieces of information provided by optimization-oriented global constraints.
What differs is the fact that in a global optimization chance-constraint we find **two stages of relaxations**.

Optimization-oriented global chance-constraints

A **global optimization chance-constraint** provides the same three pieces of information provided by optimization-oriented global constraints.
What differs is the fact that in a global optimization chance-constraint we find **two stages of relaxations**.

- At the **first stage** of relaxation we exploit known inequalities to replace stochastic variables in our stochastic programs with deterministic quantities and to yield a valid relaxation that is a deterministic problem.

Optimization-oriented global chance-constraints

A **global optimization chance-constraint** provides the same three pieces of information provided by optimization-oriented global constraints.
What differs is the fact that in a global optimization chance-constraint we find **two stages of relaxations**.

- At the **first stage** of relaxation we exploit known inequalities to replace stochastic variables in our stochastic programs with deterministic quantities and to yield a valid relaxation that is a deterministic problem.
- **A second stage of relaxation** may be needed to produce a deterministic problem that is computationally more tractable.

Value of Stochastic Solution

- Let Ξ be a discrete stochastic (vector) variable whose realizations correspond to the various **scenarios**

Value of Stochastic Solution

- Let Ξ be a discrete stochastic (vector) variable whose realizations correspond to the various **scenarios**
- Define $\mathrm{P} = \max_{x \in S} z(x, \xi)$ as the **optimization problem** associated with **one particular scenario** $\xi \in \Xi$, where S is a *finite* set, and $z(x, \xi)$ is a real valued function of two (vector) variables x and ξ

Value of Stochastic Solution

- Let Ξ be a discrete stochastic (vector) variable whose realizations correspond to the various **scenarios**
- Define $\mathrm{P} = \max_{x \in S} z(x, \xi)$ as the **optimization problem** associated with **one particular scenario** $\xi \in \Xi$, where S is a *finite* set, and $z(x, \xi)$ is a real valued function of two (vector) variables x and ξ

Recourse Problem

- RP(P)$= \max_{x \in S} \sum_{\Xi} \Pr\{\xi\} z(x, \xi)$

Value of Stochastic Solution

- Let Ξ be a discrete stochastic (vector) variable whose realizations correspond to the various **scenarios**
- Define $\mathrm{P} = \max_{x \in S} z(x, \xi)$ as the **optimization problem** associated with **one particular scenario** $\xi \in \Xi$, where S is a *finite* set, and $z(x, \xi)$ is a real valued function of two (vector) variables x and ξ

Recourse Problem

- RP(P)$= \max_{x \in S} \sum_{\Xi} \Pr\{\xi\} z(x, \xi)$
- This can be expressed, by using the **expectation operator** $\mathbb{E}$, as $\mathrm{RP(P)} = \max_{x \in S} \mathbb{E} z(x, \Xi)$, with an optimal solution x^*

Value of Stochastic Solution

Expected Value Problem

The **deterministic problem** obtained by **replacing** all the stochastic (vector) variables by their **expected values**, is defined as

$$\mathrm{EV}(\mathrm{P}) = \max_{x \in S} z(x, \mathbb{E}[\Xi]).$$

Value of Stochastic Solution

Expected Value Problem

The **deterministic problem** obtained by **replacing** all the stochastic (vector) variables by their **expected values**, is defined as

$$\mathrm{EV(P)} = \max_{x \in S} z(x, \mathbb{E}[\Xi]).$$

Let us denote by $\widehat{x}$ an optimal solution of the expected value problem, called the **expected value solution**. Anyone familiar with Stochastic Programming or realizing that uncertainty is a fact of life would feel **a little insecure about taking decision** $\widehat{x}$. Indeed, unless such a decision is independent of Ξ, there is **no reason to believe** that this decision is even **close to the optimal solution** of the recourse problem.

Value of Stochastic Solution

For any stochastic maximization (minimization) program, under the **assumptions** that

Value of Stochastic Solution

For any stochastic maximization (minimization) program, under the **assumptions** that

- (i) $z(x, \Xi)$, the profit function, is a **concave** (**convex**) function of Ξ

Value of Stochastic Solution

For any stochastic maximization (minimization) program, under the **assumptions** that

- (i) $z(x, \Xi)$, the profit function, is a **concave** (**convex**) function of Ξ
- (ii) $\max_{x \in S} z(x, \Xi)$ ($\min_{x \in S} z(x, \Xi)$) **exists** for all Ξ

Value of Stochastic Solution

For any stochastic maximization (minimization) program, under the **assumptions** that

- (i) $z(x, \Xi)$, the profit function, is a **concave** (**convex**) function of Ξ
- (ii) $\max_{x \in S} z(x, \Xi)$ ($\min_{x \in S} z(x, \Xi)$) **exists** for all Ξ

Jensen's Inequality

EV(P) - RP(P) ≥ 0 (EV(P) - RP(P) ≤ 0).

Value of Stochastic Solution

For any stochastic maximization (minimization) program, under the **assumptions** that

- (i) $z(x, \Xi)$, the profit function, is a **concave** (**convex**) function of Ξ
- (ii) $\max_{x \in S} z(x, \Xi)$ ($\min_{x \in S} z(x, \Xi)$) **exists** for all Ξ

Jensen's Inequality

EV(P) - RP(P) ≥ 0 (EV(P) - RP(P) ≤ 0).

It directly follows that EV(P) $\geq$ RP(P) (EV(P) $\leq$ RP(P)). We will base our **cost-based filtering strategies** on this **inequality**.

Expectation-based relaxation for stochastic variables

- The **first step** in our cost-based filtering strategy consists in applying a **relaxation involving the stochastic variables**.

Expectation-based relaxation for stochastic variables

- The **first step** in our cost-based filtering strategy consists in applying a **relaxation involving the stochastic variables**.
- By applying **Jensen's inequality** [EV(P) - RP(P) $\geq$ 0], if the profit (respectively cost for minimization problems) function satisfies the two assumptions discussed, an upper (lower) **bound** for the cost of an optimal solution to RP(P) can be **obtained by solving** EV(P), that is the **deterministic problem** in which all the stochastic variables are replaced by their respective **expected values**.

Expectation-based relaxation for stochastic variables

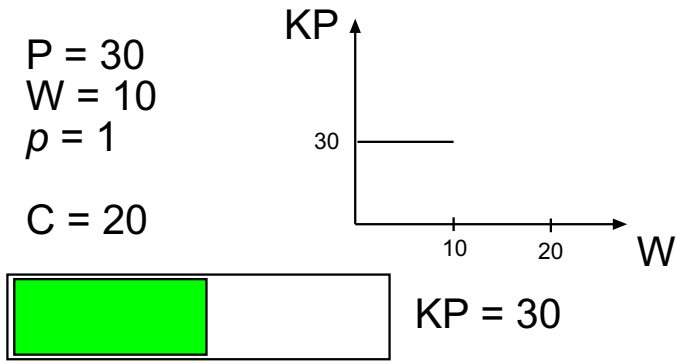
Lemma

The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.

Expectation-based relaxation for stochastic variables

Lemma

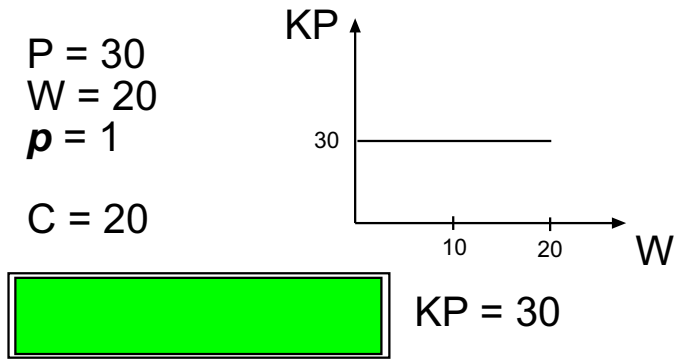
The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.



Expectation-based relaxation for stochastic variables

Lemma

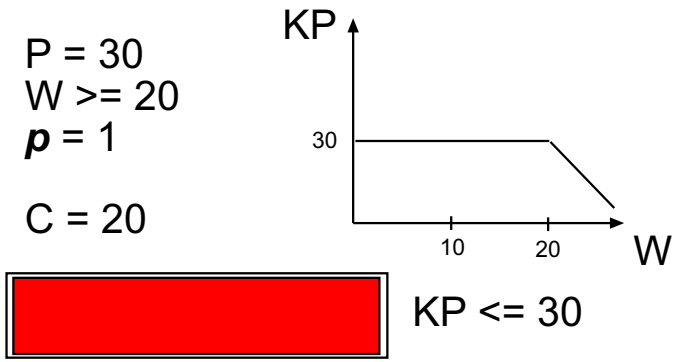
The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.



Expectation-based relaxation for stochastic variables

Lemma

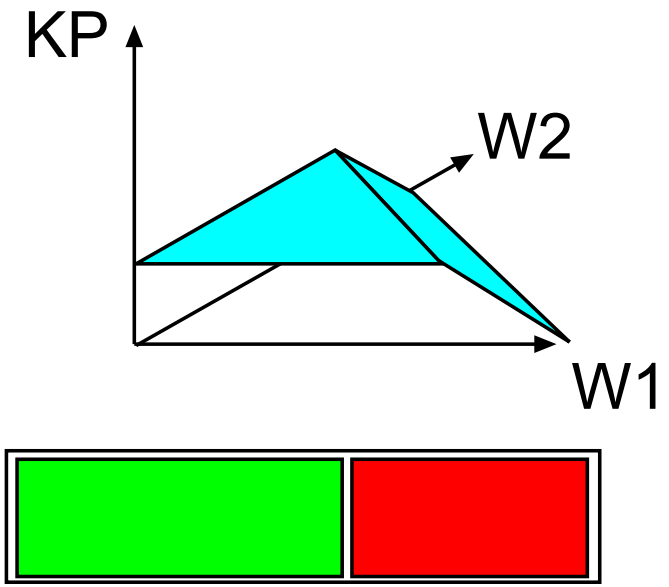
The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.



Expectation-based relaxation for stochastic variables

Lemma

The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.



Expectation-based relaxation for stochastic variables

Lemma

The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.

- Obviously, in RP(SSKP), it is always possible to find a **feasible assignment** for decision variables, therefore **both the assumptions are satisfied** for this problem.

Expectation-based relaxation for stochastic variables

Lemma

The **profit function** for $RP(SSKP)$ is **concave** in $\overline{\mathcal{W}}$.

- Obviously, in RP(SSKP), it is always possible to find a **feasible assignment** for decision variables, therefore **both the assumptions are satisfied** for this problem.
- The expected value problem EV(SSKP) can be obtained by **replacing every random variable** $\mathcal{W}_i$ in RP(SSKP) with the respective **expected value** $\mathbb{E}[\mathcal{W}_i]$, thus obtaining a fully deterministic model.

Expectation-based relaxation for stochastic variables

EV(SSKP)

Here we solve the **deterministic** problem where the weights of the objects are set to the respective **expected weights**: $\lfloor \mathbb{E}[f(1)] \rfloor = 9$, $\lfloor \mathbb{E}[f(2)] \rfloor = 11$, $\lfloor \mathbb{E}[f(3)] \rfloor = 11$, $\lfloor \mathbb{E}[f(4)] \rfloor = 5$, $\lfloor \mathbb{E}[f(5)] \rfloor = 13$.

Expectation-based relaxation for stochastic variables

EV(SSKP)

Here we solve the **deterministic** problem where the weights of the objects are set to the respective **expected weights**: $\lfloor \mathbb{E}[f(1)] \rfloor = 9$, $\lfloor \mathbb{E}[f(2)] \rfloor = 11$, $\lfloor \mathbb{E}[f(3)] \rfloor = 11$, $\lfloor \mathbb{E}[f(4)] \rfloor = 5$, $\lfloor \mathbb{E}[f(5)] \rfloor = 13$.

This problem provides the first **two pieces of information** needed by our cost-based filtering method

Expectation-based relaxation for stochastic variables

EV(SSKP)

Here we solve the **deterministic** problem where the weights of the objects are set to the respective **expected weights**: $\lfloor \mathbb{E}[f(1)] \rfloor = 9$, $\lfloor \mathbb{E}[f(2)] \rfloor = 11$, $\lfloor \mathbb{E}[f(3)] \rfloor = 11$, $\lfloor \mathbb{E}[f(4)] \rfloor = 5$, $\lfloor \mathbb{E}[f(5)] \rfloor = 13$.

This problem provides the first **two pieces of information** needed by our cost-based filtering method

- (a) the optimal solution of the **relaxed problem**

Expectation-based relaxation for stochastic variables

EV(SSKP)

Here we solve the **deterministic** problem where the weights of the objects are set to the respective **expected weights**: $\lfloor \mathbb{E}[f(1)] \rfloor = 9$, $\lfloor \mathbb{E}[f(2)] \rfloor = 11$, $\lfloor \mathbb{E}[f(3)] \rfloor = 11$, $\lfloor \mathbb{E}[f(4)] \rfloor = 5$, $\lfloor \mathbb{E}[f(5)] \rfloor = 13$.

This problem provides the first **two pieces of information** needed by our cost-based filtering method

- (a) the optimal solution of the **relaxed problem**
- (b) the **optimal value of this solution**, which represents, according to Jensen's inequality, an **upper bound** for the original problem objective function. In our running example this solution selects items $3, 4, 5$ and has a value of EV(SSKP)$= 50$.

Relaxing the expected value problem

- It should be noted that, although the expected value problem is easier than the recourse problem, it may still be **difficult to solve** (NP-hard).

Relaxing the expected value problem

- It should be noted that, although the expected value problem is easier than the recourse problem, it may still be **difficult to solve** (NP-hard).
- For this reason we can **further relax** the expected value problem in order to obtain a **valid bound** by solving an easier problem.

Relaxing the expected value problem

- It should be noted that, although the expected value problem is easier than the recourse problem, it may still be **difficult to solve** (NP-hard).
- For this reason we can **further relax** the expected value problem in order to obtain a **valid bound** by solving an easier problem.
- Let REL(EV(P)) be a **generic relaxation** of EV(P). For a maximization problem EV(P)$\leq$ REL(EV(P)), by recalling that $R(P) \leq EV(P)$, REL(EV(P)) provides a valid bound for the recourse problem, $R(P)$.

Relaxing the expected value problem

REL(EV(SSKP))

- In SSKP, for instance, **instead of solving** to optimality the deterministic (NP-Complete) knapsack problem obtained for the expected value scenario, we may instead **solve in linear time** its **continuous relaxation**, thus obtaining Dantzig's upper bound, DUB(EV(SSKP)).

Relaxing the expected value problem

REL(EV(SSKP))

- In SSKP, for instance, **instead of solving** to optimality the deterministic (NP-Complete) knapsack problem obtained for the expected value scenario, we may instead **solve in linear time** its **continuous relaxation**, thus obtaining Dantzig's upper bound, DUB(EV(SSKP)).
- DUB(EV(SSKP)) $\geq$ EV(SSKP) therefore DUB(EV(SSKP)) $\geq$ RP(SSKP).

Relaxing the expected value problem

REL(EV(SSKP))

- In SSKP, for instance, **instead of solving** to optimality the deterministic (NP-Complete) knapsack problem obtained for the expected value scenario, we may instead **solve in linear time** its **continuous relaxation**, thus obtaining Dantzig's upper bound, DUB(EV(SSKP)).
- DUB(EV(SSKP)) $\geq$ EV(SSKP) therefore DUB(EV(SSKP)) $\geq$ RP(SSKP).
- DUB(EV(SSKP)) is a **valid upper bound** for our recourse problem RP(SSKP).

Relaxing the expected value problem

SSKP

To obtain DUB(EV(SSKP))

- we **order items** for profit over expected weight: $\{25/13, 20/11, 15/11, 10/9, 5/5\}$

Relaxing the expected value problem

SSKP

To obtain DUB(EV(SSKP))

- we **order items** for profit over expected weight: $\{25/13, 20/11, 15/11, 10/9, 5/5\}$
- we **insert items** until the first that does not fit completely into the remaining knapsack capacity

Relaxing the expected value problem

SSKP

To obtain DUB(EV(SSKP))

- we **order items** for profit over expected weight: $\{25/13, 20/11, 15/11, 10/9, 5/5\}$
- we **insert items** until the first that does not fit completely into the remaining knapsack capacity
- of this last item we take **a fraction** of the profit proportional to the capacity available.

Relaxing the expected value problem

SSKP

To obtain DUB(EV(SSKP))

- we **order items** for profit over expected weight: $\{25/13, 20/11, 15/11, 10/9, 5/5\}$
- we **insert items** until the first that does not fit completely into the remaining knapsack capacity
- of this last item we take **a fraction** of the profit proportional to the capacity available.

Therefore DUB(EV(SSKP))$= 25 + 20 + (6 * 15/11) = 53.18$.

Relaxing the expected value problem

- Obviously at **any node** of the search tree it is possible to solve the **expected value problem** taking into account decision **variables already assigned**.

Relaxing the expected value problem

- Obviously at **any node** of the search tree it is possible to solve the **expected value problem** taking into account decision **variables already assigned**.
- The bound obtained can be used to **exclude part of the tree** that cannot lead to a better solution.

Relaxing the expected value problem

- Obviously at **any node** of the search tree it is possible to solve the **expected value problem** taking into account decision **variables already assigned**.
- The bound obtained can be used to **exclude part of the tree** that cannot lead to a better solution.
- In [2] the authors discuss filtering strategies based on **reduced costs** (RC). As we shall see next a similar technique can be adopted for SSKP, provided that an efficient way of obtaining bounds is available for the expected value problem.

Cost-based filtering

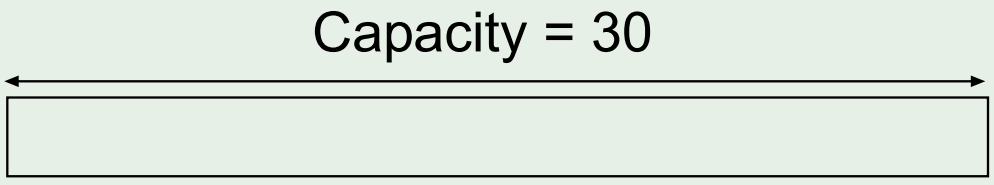
- In order to perform **cost-based filtering**, as in RC-based filtering, we need a **gradient function grad(V,v)**, which returns for each variable-value pair (V,v) an optimistic evaluation of the profit obtained if v is assigned to V.

Cost-based filtering

- In order to perform **cost-based filtering**, as in RC-based filtering, we need a **gradient function grad(V,v)**, which returns for each variable-value pair (V,v) an optimistic evaluation of the profit obtained if v is assigned to V.
- This function is obviously **problem dependent**, but **regardless of the strategy adopted** in the former section — i.e. whether we are using a relaxation for the expected value problem or solving this problem to optimality — it is possible to specify it and use it to **filter provably suboptimal values**.

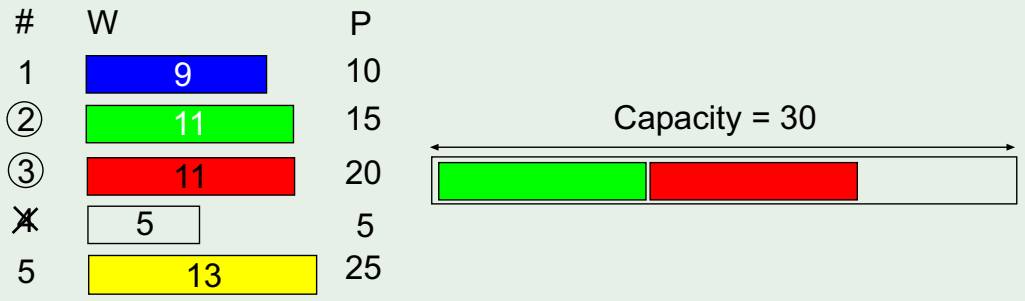
Cost-based Filtering for SSKP

#	W	P
1	9	10
2	11	15
3	11	20
4	5	5
5	13	25



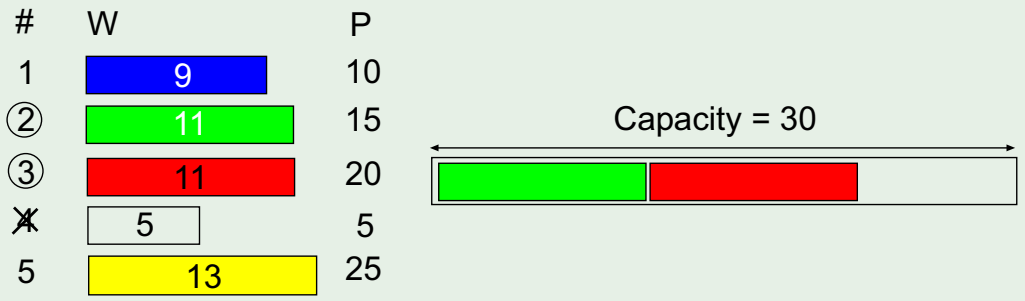
Cost-based Filtering for SSKP

We now consider the case in which items 2 and 3 have been **selected** in the knapsack and item 4 is **not selected**. We still have to **decide** on items 1 and 5.



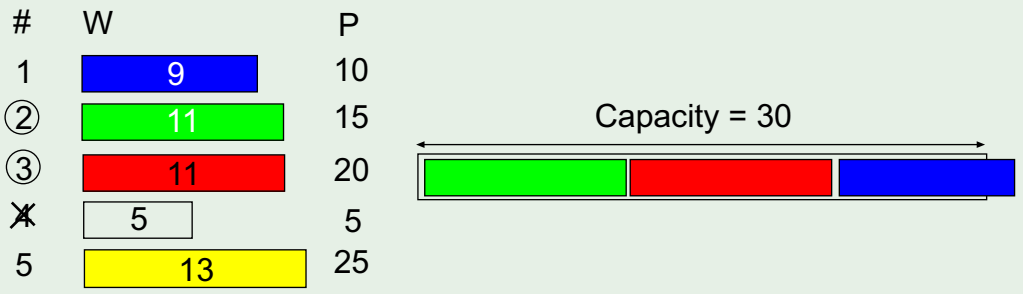
Cost-based Filtering for SSKP

We consider the set of the **remaining items** for which a decision must be taken, $K/S \equiv \{1, 5\}$.
Let us reason on item 1...



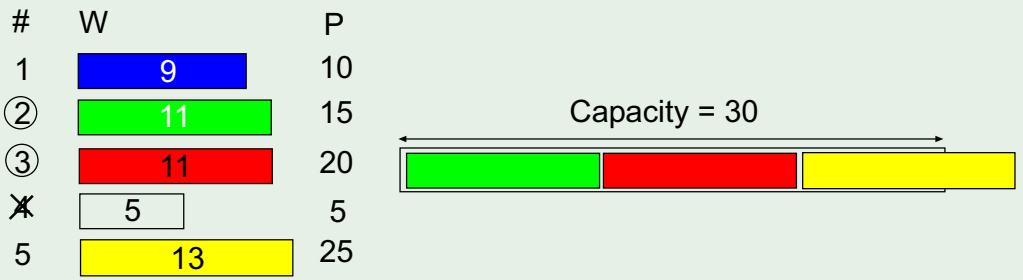
Cost-based Filtering for SSKP

Item 1 is a **critical** item, in fact if taken in the knapsack it will use more capacity than the residual $30 - 22 = 8$ units. If we consider the option of **taking this item**, then the expected profit is $\bar{r}_1 = 10 - 2 * (30 - 22 - 9) = 8$, there is no more residual capacity and item 5 is therefore excluded in the bound computation since $25 - 2 * 13 \leq 0$. The computed **bound** is $35 + 8 = 43$.



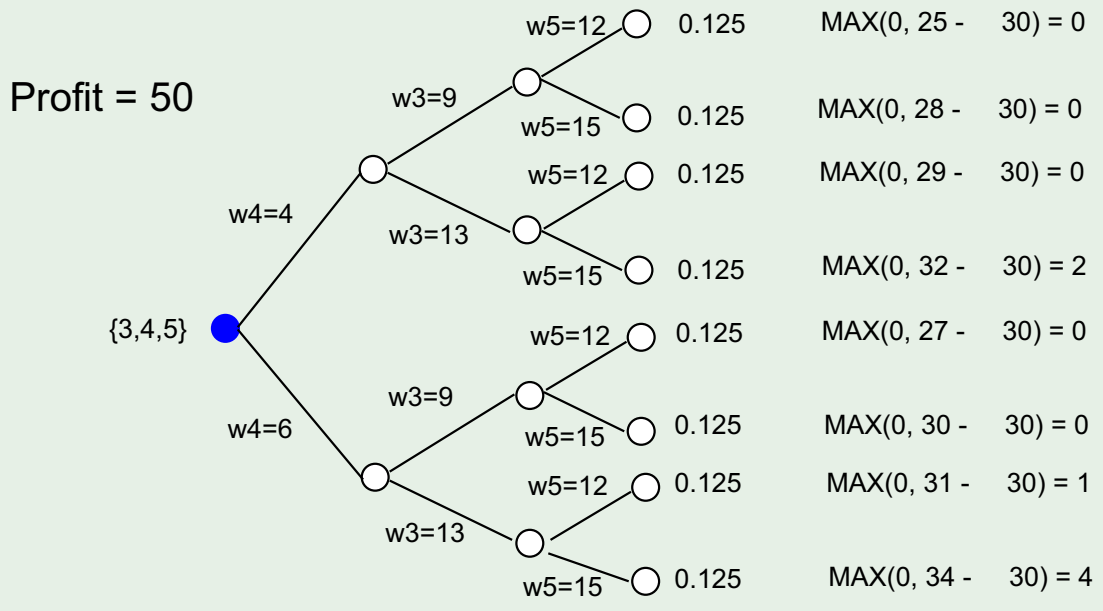
Cost-based Filtering for SSKP

The reasoning is similar for item 5. If we consider the option of **taking this item**, then the expected profit is $\overline{r}_5 = 25 - 2 * (30 - 22 - 13) = 15$, there is **no more residual capacity** and item 1 is therefore **excluded** in the bound computation since $10 - 2 * 9 \leq 0$. The computed **bound** is $35 + 15 = 50$.



Cost-based Filtering for SSKP

Assume now that the current **best solution** for RP(SSKP) has a value of 48.25, corresponding to a knapsack that contains elements 3, 4 and 5: then element 1 can be excluded from the knapsack.



Cost-based Filtering

Considerations

- Obviously, as discussed in [2] the information provided by the relaxed model (EV(P)), i.e. expected weights, gradient function etc., can be also used to **define search strategies**.

Cost-based Filtering

Considerations

- Obviously, as discussed in [2] the information provided by the relaxed model (EV(P)), i.e. expected weights, gradient function etc., can be also used to **define search strategies**.
- For instance in SSKP we may branch on variables according to a **decreasing profit over expected weight** heuristic, or selecting the one for which the **chosen gradient function gives the most promising value**.

Finding good feasible solutions

- In CP, it is critical, in order to achieve efficiency, to **quickly obtain a good feasible solution** so that cost-based filtering can **prune** provably suboptimal nodes **as early as possible**.

Finding good feasible solutions

- In CP, it is critical, in order to achieve efficiency, to **quickly obtain a good feasible solution** so that cost-based filtering can **prune** provably suboptimal nodes **as early as possible**.
- In Stochastic COPs the **EV(P) solution** can be often used as **a good starting solution** in the search process.

Finding good feasible solutions

- In CP, it is critical, in order to achieve efficiency, to **quickly obtain a good feasible solution** so that cost-based filtering can **prune** provably suboptimal nodes **as early as possible**.
- In Stochastic COPs the **EV(P) solution** can be often used as **a good starting solution** in the search process.
- **If such a solution is feasible** with respect to RP(P) — in our examples assumption (ii) guarantees this — we can easily compute EEV(P), that is *the expected result of using the EV(P) solution in the recourse problem RP(P)*.

Finding good feasible solutions

- In CP, it is critical, in order to achieve efficiency, to **quickly obtain a good feasible solution** so that cost-based filtering can **prune** provably suboptimal nodes **as early as possible**.
- In Stochastic COPs the **EV(P) solution** can be often used as **a good starting solution** in the search process.
- **If such a solution is feasible** with respect to RP(P) — in our examples assumption (ii) guarantees this — we can easily compute EEV(P), that is *the expected result of using the EV(P) solution in the recourse problem RP(P)*.
- Furthermore, at every node of the search tree it is possible to adopt a **variable fixing strategy** and compute the EV(P) solution with respect to such a node.

Finding good feasible solutions

SSKP

- In our SSKP example the **solution of the expected value problem**, EV(SSKP), selects items 3, 4 and 5 in the optimum knapsack.

Finding good feasible solutions

SSKP

- In our SSKP example the **solution of the expected value problem**, EV(SSKP), selects items 3, 4 and 5 in the optimum knapsack.
- This solution is **clearly feasible** for RP(SSKP).

Finding good feasible solutions

SSKP

- In our SSKP example the **solution of the expected value problem**, EV(SSKP), selects items 3, 4 and 5 in the optimum knapsack.
- This solution is **clearly feasible** for RP(SSKP).
- We can therefore compute EEV(SSKP)$=46$. This is, of course, a good **lower bound** for the objective function value.

Static Stochastic Knapsack Problem

Instance		Time		Nodes	
k	Scenarios	SCP	SCP-OO	SCP	SCP-OO
10	100	**0.4**	0.5	916	**100**
10	300	1.3	**0.5**	2630	**59**
10	500	2.4	**0.2**	4237	**8**
10	1000	7.2	**2.4**	6227	**120**
15	100	2.5	**0.3**	4577	**11**
15	300	15	**2.3**	10408	**252**
15	500	33	**1.1**	9982	**75**
15	1000	150	**6.3**	16957	**222**
20	100	70	**10**	102878	**1024**
20	300	250	**13**	85073	**953**
20	500	860	**9.5**	129715	**225**
20	1000	3200	**240**	134230	**7962**

Table: Experimental results for SSKP. Comparison between a pure SCP approach (SCP) and an SCP model enhanced with optimization-oriented global-chance constraints (SCP-OO), times are in seconds. In each line we indicated in bold the best performance in terms of run time and explored nodes.

Stochastic sequencing with release times and deadlines

Instance			Time		Nodes	
Jobs	Machines	Scenarios	SCP	SCP-OO	SCP	SCP-OO
3	2	10	**0.3**	**0.3**	203	**48**
3	2	30	1.3	**0.6**	701	**133**
3	2	50	3.2	**1.1**	927	**418**
3	2	100	12	**3.5**	1809	**838**
7	3	10	**180**	866	57688	**1723**
7	3	30	1800	**880**	186257	**5293**
7	3	50	3300	**1100**	212887	**6586**
7	3	100	14000	**1200**	277804	**8862**

Table: Experimental Results for SSEQ. Comparison between a pure SCP approach (SCP) and an SCP model enhanced with optimization-oriented global-chance constraints (SCP-OO), times are in seconds. In each line we indicated in bold the best performance in terms of run time and explored nodes.

Related Works

- This work extends the original work by Focacci et al. [2] on **optimization-oriented global constraints**.

Related Works

- This work extends the original work by Focacci et al. [2] on **optimization-oriented global constraints**.
- It also extends the original idea of **global chance-constraints** (Rossi et al. [4]) in such a way to incorporate cost-based reasoning.

Related Works

- This work extends the original work by Focacci et al. [2] on **optimization-oriented global constraints**.
- It also extends the original idea of **global chance-constraints** (Rossi et al. [4]) in such a way to incorporate cost-based reasoning.
- Dedicated **cost-based filtering techniques** for stochastic combinatorial optimization problems have been presented in Tarim et al. [5],

Related Works

- This work extends the original work by Focacci et al. [2] on **optimization-oriented global constraints**.
- It also extends the original idea of **global chance-constraints** (Rossi et al. [4]) in such a way to incorporate cost-based reasoning.
- Dedicated **cost-based filtering techniques** for stochastic combinatorial optimization problems have been presented in Tarim et al. [5], but these techniques are **specialized for inventory control problems**,

Related Works

- This work extends the original work by Focacci et al. [2] on **optimization-oriented global constraints**.
- It also extends the original idea of **global chance-constraints** (Rossi et al. [4]) in such a way to incorporate cost-based reasoning.
- Dedicated **cost-based filtering techniques** for stochastic combinatorial optimization problems have been presented in Tarim et al. [5], but these techniques are **specialized for inventory control problems**, while those here presented can be applied to a **wider class** of stochastic constraint programs.

Related Works

- This work builds on **known inequalities** borrowed from Stochastic Programming (see Birge & Loveaux [1]).

Related Works

- This work builds on **known inequalities** borrowed from Stochastic Programming (see Birge & Loveaux [1]).
- Stochastic Programming models are typically formulated as **dynamic programs** or **scenario-based MIP models**.

Related Works

- This work builds on **known inequalities** borrowed from Stochastic Programming (see Birge & Loveaux [1]).
- Stochastic Programming models are typically formulated as **dynamic programs** or **scenario-based MIP models**.
- In both cases these bounds are **not exploited** for **filtering** decision variable domains as in our approach and they cannot be used for **guiding the search**.

Conclusions

We proposed a novel strategy to performing **cost-based filtering** for certain classes of **stochastic constraint programs**, under the assumptions that

Conclusions

We proposed a novel strategy to performing **cost-based filtering** for certain classes of **stochastic constraint programs**, under the assumptions that

- (i) the objective function is **concave** or **convex** in the stochastic variables;

Conclusions

We proposed a novel strategy to performing **cost-based filtering** for certain classes of **stochastic constraint programs**, under the assumptions that

- (i) the objective function is **concave** or **convex** in the stochastic variables;
- (ii) the existence of a feasible solution is **not affected** by the distribution of the stochastic variables.

Conclusions

- This strategy is based on a **known inequality** (Jensen's) borrowed from Stochastic Programming.

Conclusions

- This strategy is based on a **known inequality** (Jensen's) borrowed from Stochastic Programming.
- We applied this technique to two **combinatorial optimization problems** involving uncertainty from the literature.

Conclusions

- This strategy is based on a **known inequality** (Jensen's) borrowed from Stochastic Programming.
- We applied this technique to two **combinatorial optimization problems** involving uncertainty from the literature.
- Our results confirm that **orders-of-magnitude improvements** in terms of **explored nodes** and **run times** can be achieved.

Conclusions

In the future, we aim to

- apply cost-based filtering to **multi-stage** Stochastic COPs

Conclusions

In the future, we aim to

- apply cost-based filtering to **multi-stage** Stochastic COPs
- define strategies to **handle generic chance-constraints**

Conclusions

In the future, we aim to

- apply cost-based filtering to **multi-stage** Stochastic COPs
- define strategies to **handle generic chance-constraints**
- extend the approach to **other valid inequalities** such as Edmundson-Madansky [1] or to suitable inequalities for non-convex problems [3]

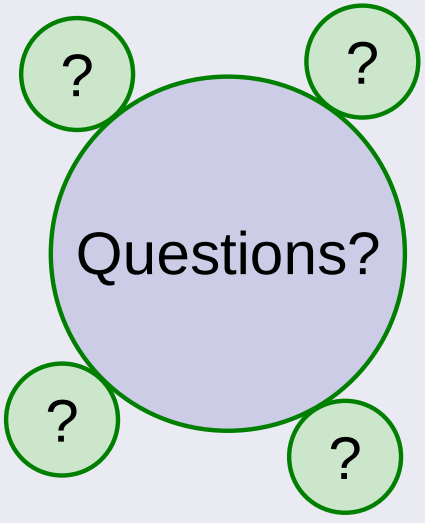
Conclusions

In the future, we aim to

- apply cost-based filtering to **multi-stage** Stochastic COPs
- define strategies to **handle generic chance-constraints**
- extend the approach to **other valid inequalities** such as Edmundson-Madansky [1] or to suitable inequalities for non-convex problems [3]
- Finally, we plan to exploit the information provided by optimization-oriented global chance-constraints to define **search strategies**

The End

The End



Acknowledgments: this work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain Research (CTVR) and Grant No. 05/IN/I886. S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027

J. R. Birge and F. Louveaux.
Introduction to Stochastic Programming.
Springer Verlag, New York, 1997.

F. Focacci, A. Lodi, and M. Milano.
Optimization-oriented global constraints.
Constraints, 7:351–365, 2002.

D. Kuhn.
Generalized bounds for convex multistage stochastic programs.
Lecture Notes in Economics and Mathematical Systems No. 584.

R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich.
A global chance-constraint for stochastic inventory systems under service level constraints.
Constraints, 13(4):xxx–xxx, 2008.

S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich.
Cost-based filtering techniques for stochastic inventory control under service level constraints.
Constraints, 2008 (forthcoming).

S. A. Tarim, S. Manandhar, and T. Walsh.
Stochastic constraint programming: A scenario-based approach.
Constraints, 11(1):53–80, 2006.

T. Walsh.
Stochastic constraint programming.
In *Proceedings of the 15th ECAI. European Conference on Artificial Intelligence*. IOS Press, 2002.